C 13

```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$2

Sy
---
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
PI(
**FILE**ID**STAPRFLNM                                    B  6




  SSSSSSSS  TTTTTTTTTT    AAAAAA    PPPPPPPP    RRRRRRRR    FFFFFFFFFF  LL          NN      NN  MM      MM
  SSSSSSSS  TTTTTTTTTT    AAAAAA    PPPPPPPP    RRRRRRRR    FFFFFFFFFF  LL          NN      NN  MM      MM
SS              TT      AA      AA  PP      PP  RR      RR  FF          LL          NN      NN  MMMM  MMMM
SS              TT      AA      AA  PP      PP  RR      RR  FF          LL          NN      NN  MMMM  MMMM
SS              TT      AA      AA  PP      PP  RR      RR  FF          LL          NNNN    NN  MM  MM  MM
SS              TT      AA      AA  PP      PP  RR      RR  FF          LL          NNNN    NN  MM  MM  MM
  SSSSSS        TT      AA      AA  PPPPPPPP    RRRRRRRR    FFFFFFFF    LL          NN  NN  NN  MM      MM
  SSSSSS        TT      AA      AA  PPPPPPPP    RRRRRRRR    FFFFFFFF    LL          NN  NN  NN  MM      MM
        SS      TT      AAAAAAAAAA  PP          RR  RR      FF          LL          NN    NNNN  MM      MM
        SS      TT      AAAAAAAAAA  PP          RR  RR      FF          LL          NN    NNNN  MM      MM
        SS      TT      AA      AA  PP          RR    RR    FF          LL          NN      NN  MM      MM     ....
        SS      TT      AA      AA  PP          RR    RR    FF          LL          NN      NN  MM      MM     ....
SSSSSSSS        TT      AA      AA  PP          RR      RR  FF          LLLLLLLLLL  NN      NN  MM      MM     ....
SSSSSSSS        TT      AA      AA  PP          RR      RR  FF          LLLLLLLLLL  NN      NN  MM      MM     ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
    (3)      221        DECLARATIONS
    (5)      413        RMS$ASSIGN, Assign a Channel
    (6)      585        GETDEV_CHAR - Determine the device characteristics
    (15)    1070        RMS$RET_DEV_CHAR - Set device characteristics into FAB
```

```
00000001  0000     1 STASWITCH=1                                  ;GENERATE STAND-ALONE RMS CODE FOR
          0000     2                                              ; USE IN STAND-ALONE BACKUP, ET. AL.
          0000     1         .IF     DF,STASWITCH
          0000     2         $BEGIN  STAPRFLNM,000,RM$RMS0,<PROCESS FILE NAME>
          0000     3         .IFF    ;STASWITCH
          0000     4         $BEGIN  RMOPRFLNM,033,RM$RMS0,<PROCESS FILE NAME>
          0000     5         .ENDC   ;STASWITCH
          0000     6
          0000     7 ;
          0000     8 ;****************************************************************************
          0000     9 ;*                                                                          *
          0000    10 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
          0000    11 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
          0000    12 ;*  ALL RIGHTS RESERVED.                                                    *
          0000    13 ;*                                                                          *
          0000    14 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
          0000    15 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
          0000    16 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
          0000    17 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
          0000    18 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
          0000    19 ;*  TRANSFERRED.                                                            *
          0000    20 ;*                                                                          *
          0000    21 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
          0000    22 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
          0000    23 ;*  CORPORATION.                                                            *
          0000    24 ;*                                                                          *
          0000    25 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
          0000    26 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
          0000    27 ;*                                                                          *
          0000    28 ;*                                                                          *
          0000    29 ;****************************************************************************
          0000    30 ;
```

```
0000   ;++
0000   ; Facility: RMS32
0000   ;
0000   ; Abstract:
0000   ;               This routine performs RMS32 file name processing.
0000   ;
0000   ; Environment:
0000   ;               Star processor running Starlet exec.
0000   ;
0000   ; Author: L. F. Laverdure          Creation Date: 4-JAN-1977
0000   ;
0000   ; Modified By:
0000   ;
0000   ;       V03-033 RAS0329         Ron Schaefer            31-Jul-1984
0000   ;               Fix RM$PRFLNM to set GET access internally if EXE access is
0000   ;               requested; this is needed for execute-only command procedures.
0000   ;
0000   ;       V03-032 RAS0296         Ron Schaefer            18-Apr-1984
0000   ;               Add secondary device name item code to $GETDVI.
0000   ;               This name is returned in NAM$T_DVI if the device is spooled.
0000   ;
0000   ;       V03-031 RAS0289         Ron Schaefer            6-Apr-1984
0000   ;               Add notranslation flag to call to $ASSIGN; change
0000   ;               to WLD error when wildcard detected.
0000   ;
0000   ;       V03-030 RAS0268         Ron Schaefer            12-Mar-1984
0000   ;               Move searchlist loop decision to caller of RM$PRFLNM[ALT].
0000   ;
0000   ;       V03-029 DGB0026         Donald G. Blair         11-Mar-1984
0000   ;               Change BSBW to JSB to fix broken branch.
0000   ;
0000   ;       V03-028 RAS0242         Ron Schaefer            23-Jan-1984
0000   ;               Fix bugchecks caused by lack of a fwa (valid R10)
0000   ;               on calls to RM$PRFLNM or error returns from RM$XPFN.
0000   ;
0000   ;       V03-027 RAS0229         Ron Schaefer            5-Jan-1984
0000   ;               Return an error if $OPEN/$CREATE specifies a directory
0000   ;               containing "..." after expansion.   Related file
0000   ;               processing will resolve ellipses normally.
0000   ;
0000   ;       V03-026 RAS0225         Ron Schaefer            20-Dec-1983
0000   ;               Fix stream PPF file with FTN carriage control problem;
0000   ;               eliminate setting of NAM$V_CNCL_DEV and NAM$V_ROOT_DIR
0000   ;               in RM$ASSIGN as this is now done by RM$EXPSTRING.
0000   ;
0000   ;       V03-025 RAS0212         Ron Schaefer            15-Nov-1983
0000   ;               Fix version/implementation skew in RM$ASSIGN by
0000   ;               conditionalizing this module for both RMS and
0000   ;               stand-alone BACKUP on the STASWITCH parameter.
0000   ;
0000   ;       V03-024 RAS0209         Ron Schaefer            4-Nov-1983
0000   ;               Clean-up returned device characteristics by defining
0000   ;               a routine RM$RET_DEV_CHAR to do a uniform job.
0000   ;
0000   ;       V03-023 RAS0198         Ron Schaefer            6-Oct-1983
0000   ;               Change RM$PRFLNM[ALT] logic to not have an input parameter.
0000   ;
```

```
0000     89 ;      V03-022 SHZ0001         Stephen H. Zalewski     12-Sep-1983
0000     90 ;              Add a new itemcode to the itemcode list for the $GETDVI.
0000     91 ;
0000     92 ;      V03-021 RAS0183         Ron Schaefer             2-Sep-1983
0000     93 ;              Make searchlists be non-wildcard characters.
0000     94 ;
0000     95 ;      V03-020 KBT0560         Keith B. Thompson       21-Jul-1983
0000     96 ;              Set name block flags correctly
0000     97 ;
0000     98 ;      V03-019 SHZ0001         Stephen H. Zalewski     26-Jun-1983
0000     99 ;              Get full device name (node$device_name) from GETDVI.
0000    100 ;
0000    101 ;      V03-018 KBT0538         Keith B. Thompson       6-Jun-1983
0000    102 ;              Turn on search list
0000    103 ;
0000    104 ;      V03-017 KBT0515         Keith B. Thompson       23-May-1983
0000    105 ;              RM$XPFN moved
0000    106 ;
0000    107 ;      V03-016 KBT0510         Keith B. Thompson       5-May-1983
0000    108 ;              Use RM$DEALLOCATE_FWA add some search list stuff and
0000    109 ;              redo rooted directories
0000    110 ;
0000    111 ;      V03-015 RAS0146         Ron Schaefer            18-Apr-1983
0000    112 ;              Fix FWA$C_FIBLEN assume to have slightly more
0000    113 ;              tolerance for ACP FIB length changes.
0000    114 ;
0000    115 ;      V03-014 RAS0143         Ron Schaefer            11-Apr-1983
0000    116 ;              Fix device name buffer length to be 16, and change
0000    117 ;              $GETDVI to $GETDVIW.
0000    118 ;
0000    119 ;      V03-013 RAS0139         Ron Schaefer            24-Mar-1983
0000    120 ;              Undo RAS0129.  Not returning the true RAT and RFM values
0000    121 ;              is a "bug", since many programs do not set any values
0000    122 ;              on $OPEN and expect to find out what the RAT and RFM values
0000    123 ;              are from RMS.
0000    124 ;
0000    125 ;      V03-012 RAS0129         Ron Schaefer            4-Mar-1983
0000    126 ;              Do not destroy the user's RAT and RFM values for
0000    127 ;              indirect PPFs.
0000    128 ;
0000    129 ;      V03-011 JWH0187         Jeffrey W. Horn         15-Feb-1983
0000    130 ;              Add volume lable to list of items retrieved with
0000    131 ;              $GETDVI.
0000    132 ;
0000    133 ;      V03-010 KBT0485         Keith B. Thompson       4-Feb-1983
0000    134 ;              Deallocate all pages off the ifab free list
0000    135 ;
0000    136 ;      V03-009 KBT0473         Keith B. Thompson       24-Jan-1983
0000    137 ;              Fix kbt0469, provide the correct size to RM$RETSPC.
0000    138 ;
0000    139 ;      V03-008 KBT0469         Keith B. Thompson       24-Jan-1983
0000    140 ;              Deallocate the fwa on indirect ppf open
0000    141 ;
0000    142 ;      V03-007 KBT0432         Keith B. Thompson       3-Dec-1982
0000    143 ;              Change the way the device name is stored in shrfilbuf
0000    144 ;
0000    145 ;      V03-006 KBT0407         Keith B. Thompson       10-Nov-1982
```

```
0000    146 ;               Fix a broken assume (from unfolding fwa) and call new
0000    147 ;               $getdvi system service
0000    148 ;
0000    149 ;       V03-005 KBT0212         Keith B. Thompson       23-Aug-1982
0000    150 ;               Reorganize psects
0000    151 ;
0000    152 ;       V03-004 KBT0098         Keith B. Thompson       13-Jul-1982
0000    153 ;               Clean up psects
0000    154 ;
0000    155 ;       V03-003 KEK0026         K. E. Kinnear           23-Mar-1982
0000    156 ;               More work like KEK0022 -- correctly allow foreign devices
0000    157 ;               to have ANSI-"a" names. Also add back in NFS to FOREIGN
0000    158 ;               branch deleted by RAS0067.
0000    159 ;
0000    160 ;       V03-002 RAS0079         Ron Schaefer            17-Mar-1982
0000    161 ;               Add translation table mask support (FAB$B_DSBMSK).
0000    162 ;
0000    163 ;       V03-001 RAS0080         Ron Schaefer            17-Mar-1982
0000    164 ;               Correct stream format carriage control for indirect PPFs,
0000    165 ;               both terminals and files.
0000    166 ;
0000    167 ;       V02-044 RAS0067         Ron Schaefer            9-Feb-1982
0000    168 ;               Re-arrange execution paths so that FWA$T_SHRFILDEV
0000    169 ;               is filled in for all devices.  This field is used to
0000    170 ;               return a canonical device name in NAM$T_DVI.
0000    171 ;
0000    172 ;       V02-043 KEK0022         K. E. Kinnear           2-Feb-1982
0000    173 ;               Open restrictions on quoted strings to all non-disk
0000    174 ;               (actually non-DIR) devices.
0000    175 ;
0000    176 ;       V02-042 KEK0018         K. E. Kinnear           18-Jan-1982
0000    177 ;               Modifications to allow ANSI-"a" filespecs only on
0000    178 ;               magtape devices.  Also, remove all references to
0000    179 ;               NAM$x_QUOTED and replace with NAM$x_NAME.
0000    180 ;
0000    181 ;       V02-041 RAS0060         Ron Schaefer            15-Jan-1982
0000    182 ;               Fix temp buffer usage for concealed devices;
0000    183 ;               force stream PPF terminals to have RAT=CR.
0000    184 ;
0000    185 ;       V02-040 RAS0058         Ron Schaefer            8-Jan-1982
0000    186 ;               Correct concealed device error path so as to return
0000    187 ;               $ASSIGN errors; and fix bad code saving FWA$T_SHRFILDEV.
0000    188 ;
0000    189 ;       V02-039 RAS0047         Ron Schaefer            18-Nov-1981
0000    190 ;               Fix the UIC form of directories under rooted directories,
0000    191 ;               in order to make the AME and compatibility mode work.
0000    192 ;
0000    193 ;       V02-038 KPL0002         Peter Lieberwirth       15-Nov-1981
0000    194 ;               Fix RAS0045 to not use FWA$T_XLTBUFF1 in RM$ASSIGN.
0000    195 ;               Use unnamed space at end of first page instead.
0000    196 ;               (Fix this better ASAP!)
0000    197 ;
0000    198 ;       V02-037 RAS0045         Ron Schaefer            11-Nov-1981
0000    199 ;               Complete KPL0001 by changing the reference from
0000    200 ;               FWA$T_WILD to FWA$T_SHRFILDEV.  Also fix bad register
0000    201 ;               reference.
0000    202 ;
```

```
0000  ;       V02-036 RAS0040         Ron Schaefer            16-Oct-1981
0000  ;               Implement rooted directories for concealed devices.
0000  ;               Restructure RM$ASSIGN to parse the translation of the
0000  ;               concealed device name, assign a channel to the concealed
0000  ;               device and set the internal MFD to the DID of the
0000  ;               root directory.
0000  ;
0000  ;       V02-035 KPL0001         P Lieberwirth           7-Oct-1981
0000  ;               Save unit number and device name in FWA$T_WILD field for
0000  ;               later use in naming shared file.
0000  ;
0000  ;       V02-034 RAS0025         Ron Schaefer            18-Aug-1981
0000  ;               Allow $GET/$PUT for UDF seq org files, except as
0000  ;               indirect PPF.
0000  ;
0000  ;--
0000
```

```
                0000          .SBTTL  DECLARATIONS
                0000
                0000    ;
                0000    ; Include Files:
                0000    ;
                0000
                0000    ;
                0000    ; Macros:
                0000    ;
                0000
                0000          $ASSIGNDEF
                0000          $DEVDEF
                0000          $DVIDEF
                0000          $FABDEF
                0000          $FIBDEF
                0000          $FSCBDEF
                0000          $FWADEF
                0000          $IFBDEF
                0000          $IMPDEF
                0000          $NAMDEF
                0000          $PSLDEF
                0000          $RMSDEF
                0000
                0000          .IF     DF,STASWITCH
                0000          $CCBDEF
                0000          $DCDEF
                0000          $UCBDEF
                0000          .ENDC   ;STASWITCH
                0000
                0000    ;
                0000    ; Equated Symbols:
                0000    ;
                0000
    00000020    0000          FOP=FAB$L_FOP*8                 ; bit offset to fop
                0000
                0000    ;
                0000    ; Own Storage:
                0000    ;
                0000
```

```
0000        .IF     NDF,STASWITCH
0000        .SBTTL  RM$PRFLNM, Filename Processing Routine
0000
0000  ;++
0000  ;
0000  ; RM$PRFLNM -- Filename Processing Routine.
0000  ;
0000  ;  This routine first sets up various file access and sharing bits,
0000  ;  then allocates a buffer and bdb to use as a scratch work area for building
0000  ;  the expanded filename string and interfacing with f11acp, calls rm$xpfn
0000  ;  to expand the filename string, assigns an i/o channel, and finally
0000  ;  retrieves the device characteristics, filling in the associated
0000  ;  IFAB and FAB fields.
0000  ;
0000  ;  If the result of the file name expansion indicates that the file in
0000  ;  question is an indirectly accessed process permanent file, no i/o
0000  ;  channel need be assigned, as this has already been done.  Instead,
0000  ;  an ifi is constructed that points this fab at the associated process
0000  ;  permanent file.  This has the side effect of turning a $create call for
0000  ;  an indirectly accessed process permanent file into an $open.  This is
0000  ;  done by returning to the $OPEN code rather than the $CREATE code.
0000  ;
0000  ;  If an error occurs, cleanup may be required to
0000  ;  deallocate the bdb and buffer and deassign the channel.
0000  ;
0000  ; Calling Sequence:
0000  ;
0000  ;       BSBW    RM$PRFLNM
0000  ;
0000  ; Input Parameters:
0000  ;
0000  ;       R11     impure area address
0000  ;       R10     IFAB address
0000  ;       R9      IFAB address
0000  ;       R8      FAB address
0000  ;       R0      input error status (only if search list operation)
0000  ;
0000  ; Implicit Inputs:
0000  ;
0000  ;       The contents of the various FAB and IFAB fields
0000  ;       (see functional spec for details), in particular,
0000  ;       the various file name specification fields.
0000  ;
0000  ; Output Parameters:
0000  ;
0000  ;       R10     FWA address
0000  ;       R0      status code (could be R0 input status)
0000  ;       R1 thru R5 destroyed
0000  ;       none
0000  ;
0000  ; Implicit Outputs:
0000  ;
0000  ;       Various fab, fwa, and ifab fields are filled in (see
0000  ;       functional spec for details).
0000  ;       FWA$Q_FIB initialized.
0000  ;       device fields in the ifab filled in.
0000  ;
```

```
0000       ; Completion Codes:
0000       ;
0000       ;       Standard rms, in particular, success, dev,
0000       ;       chn, and dme, in addition to the codes returned
0000       ;       by RM$XPFN.
0000       ;
0000       ; Side Effects:
0000       ;
0000       ;       See note above on change of $create into $open for indirect ppf.
0000       ;
0000       ;--
0000
0000       RM$PRFLNM::
0000
0000       ;
0000       ;  Fill in fac IFAB field from fab handling defaults and setting summary
0000       ;  write access bit as required.
0000       ;
0000
0000               MOVB    FAB$B_FAC(R8),IFB$B_FAC(R9)
0000               BEQL    SETGET                          ; branch if null
0000               BBS     #FAB$V_EXE,IFB$B_FAC(R9),SETGET ; or if EXE desired
0000
0000       ;
0000       ;  Entry point for RMS0CREATE (IFB$B_FAC already set)
0000       ;
0000
0000       RM$PRFLNMALT::
0000               BITB    #FAB$M_UPD!FAB$M_DEL!FAB$M_TRN,-
0000                       IFB$B_FAC(R9)
0000               BEQL    SETWRT                          ; branch if not upd, del, or trn
0000                                                       ; accessed, else imply get access.
0000       SETGET: BISB2   #FAB$M_GET,IFB$B_FAC(R9)        ; default to get access
0000       SETWRT: BITB    #FAB$M_PUT!FAB$M_UPD!FAB$M_DEL!FAB$M_TRN,IFB$B_FAC(R9)
0000               BEQL    20$                             ; branch if none of the varieties
0000                                                       ; of write access is specified.
0000               SSB     #IFB$V_WRTACC,(R9)              ; set summary bit
0000
0000       ;
0000       ;  Go expand file name.
0000       ;
0000
0000       20$:    JSB     RM$XPFN                         ; get fully qualified file name
0000               BLBC    R0,60$                          ; quit on error
0000
0000       ;
0000       ; Check for residual wild characters in file name.
0000       ; If FWA$V_QUOTED, then skip check.  This can either be ANSI-"a" filespecs
0000       ; where FWA$Q_NAME really holds a quoted string, or it could be a network
0000       ; quoted string, where there is nothing in FWA$Q_NAME.
0000       ;
0000
0000               BBC     #FWA$V_WILDCARD,(R10),50$       ; don't check if no wilds
0000               BBS     #FWA$V_QUOTED,(R10),40$         ; don't check if quoted
0000               LOCC    #^A'*',FWA$Q_NAME(R10),-        ; any *'s in name?
0000                       @FWA$Q_NAME+4(R10)
0000               BNEQ    90$                             ; if neq yes
```

```
0000             LOCC    #^A'%',FWA$Q_NAME(R10),-              ; any %'s in name
0000                     @FWA$Q_NAME+4(R10)
0000             BNEQ    90$                                   ; if neq yes
0000
0000     ;
0000     ; Check for residual ellipses in the directory spec.   Related file
0000     ; processing should resolve valid uses previously.   Unfortunately,
0000     ; no simple check is possible -- one has to look at each directory
0000     ; spec for the FSCB$V_ELIPS bit.
0000     ;
0000
0000     40$:    BBC     #FWA$V_WILD_DIR,(R10),50$             ; don't bother if no wild dirs
0000             EXTZV   #FWA$V_DIR_LVLS,#FWA$S_DIR_LVLS,-; get # of dirs to check
0000                     (R10),R0                              ;
0000             MOVAQ   FWA$Q_DIR1(R10),R1                    ; ptr to dir descriptors
0000     45$:    BBS     #FSCB$V_ELIPS,(R1),100$               ; ... is a no-no
0000             ADDL2   #8,R1                                 ; next directory
0000             SOBGEQ  R0,45$                                ; while they last
0000
0000     ;
0000     ;  Check for indirect open of process permanent file.
0000     ;
0000
0000     50$:    TSTB    FWA$B_ESCFLG(R10)                     ; did we get a ppf flag?
0000             BNEQ    80$                                   ; branch if so
0000             BSBB    RM$ASSIGN                             ; assign a channel
0000     60$:    RSB                                           ; exit
0000
0000     80$:    BRW     INDPPF                                ; process indirect ppf file
0000
0000     90$:    RMSERR  WLD                                   ; give wildcard file name error
0000             RSB
0000
0000     100$:   RMSERR  DIR                                   ; give bad directory error
0000             RSB
0000
0000             .ENDC   ;STASWITCH
```

```
                     0000          .SBTTL  RM$ASSIGN, Assign a Channel
                     0000  ;++
                     0000  ;
                     0000  ; RM$ASSIGN
                     0000  ;
                     0000  ;  Now assign a channel and get the associated device characteristics.
                     0000  ;
                     0000  ;  Assign in MAX( requested mode,caller's mode ) if NFS or UFO set,
                     0000  ;  else assign in exec mode.
                     0000  ;
                     0000  ;--
                     0000
                     0000  RM$ASSIGN::
                     0000
                     0000          .IF     NDF,STASWITCH
                     0000
                     0000          BBS     #FWA$V_NODE,(R10),NTASGN        ; branch if network request
                     0000
                     0000          .ENDC   ;STASWITCH
                     0000
                     0000
                     0000  ;
                     0000  ; Note: Device name already has been prefixed with an underscore, so
                     0000  ;       assign system service will not attempt to translate it again.
                     0000  ;
                     0000  ASSIGN:
            00   DD  0000          PUSHL   #0                      ; no mailbox
                     0002
                     0002          ASSUME  ASSIGN$_MBXNAM  EQ      ASSIGN$_ACMODE+4
                     0002
            01   DD  0002          PUSHL   #PSL$C_EXEC             ; exec mode
                     0004
                     0004          ASSUME  FAB$V_UFO       GE      16
                     0004          ASSUME  FAB$V_NFS       GE      16
                     0004
   06 A8    03   93  0004          BITB    #<FAB$M_UFO!FAB$M_NFS>@-16,FAB$L_FOP+2(R8)
            10   13  0008          BEQL    40$                     ; branch if neither ufo nor nfs set
            02   EF  000A          EXTZV   #FAB$V_CHAN_MODE,-      ; replace with requested mode
            02       000C                  #FAB$S_CHAN_MODE,-      ;
   6E    4A A8       000D                  FAB$B_ACMODES(R8),(SP)  ;
   6E    0A A9   91  0010          CMPB    IFB$B_MODE(R9),(SP)     ; compare with caller's mode
            04   1B  0014          BLEQU   40$                     ; maximize
   6E    0A A9   9A  0016          MOVZBL  IFB$B_MODE(R9),(SP)     ; switch to caller's mode
                     001A
                     001A          ASSUME  ASSIGN$_CHAN    EQ      ASSIGN$_ACMODE-4
                     001A
                     001A  40$:    ssb     #7,(sp)                 ; set notranslate flag
      20 A9      DF  001E          PUSHAL  IFB$W_CHNL(R9)          ; get channel # back here
                     0021
                     0021          ASSUME  ASSIGN$_DEVNAM  EQ      ASSIGN$_CHAN-4
                     0021
    00E0 CA      7F  0021          PUSHAQ  FWA$Q_DEVICE(R10)       ; device name descriptor
         39      E1  0025          BBC     #FWA$V_CONCEAL_DEV,-    ; is this a concealed device
      05 6A          0027                  (R10),45$               ;
    00E8 CA      7E  0029          MOVAQ   FWA$Q_CONCEAL_DEV(R10),-; yes replace the descriptor
         6E          002D                  (SP)                    ;
                     002E
```

```
                         002E              ASSUME  ASSIGN$_NARGS    EQ       4
                         002E
      00000000'9F  04  FB 002E     45$:    CALLS   #4,@#SYS$ASSIGN             ; and do the assign
               63 50   E9 0035             BLBC    R0,ERRASGN
                         0038
                         0038     ;
                         0038     ;  Assign succeeded  -  get and set device characteristics.
                         0038     ;
                         0038
                 0092  30 0038             BSBW    GETDEV_CHAR
                         003B
                         003B              .IF     NDF,STASWITCH
                         003B
                         003B              BLBC    (R11),DIRECTPPF1
                         003B
                         003B     ;
                         003B     ; Set up fib descriptor.
                         003B     ;
                         003B
                         003B     RM$SETFIB::
                         003B
                         003B              ASSUME  FIB$C_LENGTH     LE      FWA$C_FIBLEN
                         003B
                         003B              MOVZWL  #FIB$C_LENGTH,FWA$Q_FIB(R10)      ; set length of fib
                         003B              MOVAB   FWA$T_FIBBUF(R10),-               ; set address of fib buffer
                         003B                      FWA$Q_FIB+4(R10)                  ;
                         003B
                         003B     ERROR:   RSB
                         003B
                         003B     DIRECTPPF1:
                         003B              BRW     DIRECTPPF                   ; branch if process-perm file
                         003B
                         003B              .ENDC   ;STASWITCH
                         003B
                         003B              .IF     DF,STASWITCH
                         003B
   49 00000000'EF  00'  E0 003B             BBS     S^#EXE$V_INIT,EXE$GL_FLAGS,110$ ; br if running online
         01    1C AA   91 0043             CMPB    FWA$L_DEV_CLASS(R10),#DC$_DISK  ; disk device?
                  06   13 0047             BEQL    100$                            ; br if disk
         02    1C AA   91 0049             CMPB    FWA$L_DEV_CLASS(R10),#DC$_TAPE  ; tape device?
                  3D   12 004D             BNEQ    110$                            ; br if not tape
                          004F     100$:   SSB     #DEV$V_FOR,IFB$L_PRIM_DEV(R9) ; simulate mount/foreign
                          0053             SSB     #DEV$V_FOR,IFB$L_AS_DEV(R9) ; simulate mount/foreign
                          0059             $CMKRNL_S       STA_VOL_VAL     ; set volume valid
               30 50   E9 0068             BLBC    R0,ERRASGN                      ; br if verifychan failed
                          006B             $QIOW_S -
                          006B                     FUNC = #IO$_PACKACK,-
                          006B                     EFN = #IMP$C_ASYQIOEFN,-        ; throw-away efn
                          006B                     CHAN = IFB$W_CHNL(R9)
               0F 50   E9 0089             BLBC    R0,ERRASGN                      ; br if QIO failed
                          008C     110$:   $DASSGN_S       CHAN=IFB$W_CHNL(R9)
               20 A9   B4 0097             CLRW    IFB$W_CHNL(R9)                  ; clear channel number
                       05 009A             RSB                                     ; exit
                          009B
                          009B             .ENDC   ;STASWITCH
                          009B
                          009B             .IF     NDF,STASWITCH
```

```
                              009B  ;++
                              009B  ;
                              009B  ;  Assign a channel to the network device and get and set its device
                              009B  ;  characteristics.
                              009B  ;
                              009B  ;--
                              009B
                              009B  NTASGN:
                              009B          JSB     NT$ASSIGN               ; assign a channel
                              009B          BLBC    R0,ERRASGN              ; branch on error
                              009B          BSBB    GETDEV_CHAR             ; get network device characteristics
                              009B          JMP     NT$MOD_DEV_CHAR         ; modify device characteristics
                              009B                                          ;  and return to caller
                              009B          .ENDC   ;STASWITCH
                              009B
                              009B  ;
                              009B  ;  Handle assign error.
                              009B  ;
                              009B
                              009B  ERRASGN:
                              009B
                              009B          .IF     NDF,STASWITCH
                              009B
                              009B          RMSERR  CHN,R1                  ; unspecified chnl assgn failure
                              009B          BRW     RM$MAPERR               ; map error and return to caller
                              009B
                              009B          .ENDC   ;STASWITCH
                              009B
                              009B          .IF     DF,STASWITCH
                              009B
      51    24 A9       D0    009B          MOVL    IFB$L_LAST_FAB(R9),R1   ; get FAB address
   0C A1       50       D0    009F          MOVL    R0,FAB$L_STV(R1)        ; set STV value
                              00A3          RMSERR  CHN,R0                  ; set R0
                        05    00A8          RSB                             ; exit
                              00A9
                              00A9          .ENDC   ;STASWITCH
                              00A9
                              00A9  ;
                              00A9  ; Kernel mode routine to set volume valid and mark device mounted
                              00A9  ; for use in stand-alone Backup.
                              00A9  ;
                              00A9
                              00A9          .IF     DF,STASWITCH
                              00A9
                              00A9  STA_VOL_VAL:
                    003C      00A9          .WORD   ^M<R2,R3,R4,R5>         ; kernel mode routine to set valid
      50    20 A9       3C    00AB          MOVZWL  IFB$W_CHNL(R9),R0       ; load channel number
   00000000'9F          16    00AF          JSB     @#IOC$VERIFYCHAN        ; get CCB address for channel
         14 50          E9    00B5          BLBC    R0,210$                 ; br if failed
         51 61          D0    00B8          MOVL    CCB$L_UCB(R1),R1        ; get UCB address from CCB
   64 A1    0800 8F     A8    00BB          BISW2   #UCB$M_VALID,UCB$W_STS(R1)      ; set software valid bit
38 A1    01080000 8F    C8    00C1          BISL2   #DEV$M_MNT!DEV$M_FOR,UCB$L_DEVCHAR(R1)
                              00C9                                          ; set mounted foreign bits
         50    01       D0    00C9          MOVL    #1,R0                   ; indicate success
                        04    00CC  210$:   RET                             ; return
                              00CD
                              00CD          .ENDC   ;STASWITCH
                        00CD   585            .SBTTL  GETDEV_CHAR - Determine the device characteristics
                        00CD   586    ;
                        00CD   587    ; Subroutine to get and set device characteristics.
                        00CD   588    ;
                        00CD   589    ;       R0-R2   destroyed
                        00CD   590    ;
                        00CD   591    ;
                        00CD   592    ; The device characteristics wanted are:
                        00CD   593    ;
                        00CD   594    ;       DVI$_DEVCHAR              =>      IFB$L_PRIM_DEV
                        00CD   595    ;       DVI$_DEVCHAR (sec)        =>      IFB$L_AS_DEV
                        00CD   596    ;       DVI$_DEVBUFSIZ            =>      IFB$L_DEVBUFSIZ
                        00CD   597    ;       DVI$_DEVBUFSIZ (sec)      =>      IFB$L_ASDEVBSIZ
                        00CD   598    ;       DVI$_DEVCLASS             =>      FWA$L_DEVCLASS
                        00CD   599    ;       DVI$_FULLDEVNAM           =>      FWA$T_SHRFILBUF (pointed to by shrfil)
                        00CD   600    ;       DVI$_FULLDEVNAM (sec)     =>      FWA$T_AS_SHRFILBUF (pointed to by as_shrfil)
                        00CD   601    ;       DVI$_DEVLOCKNAM           =>      FWA$T_SHRFIL_LCKNAM (pointed to by shrfil_lc
                        00CD   602    ;       DVI$_RECSIZ               =>      FWA$L_RECSIZ
                        00CD   603    ;
                        00CD   604
                        00CD   605    GETDEV_CHAR:
                        00CD   606
                        00CD   607    ;
                        00CD   608    ; Create item list on stack
                        00CD   609    ;
       52    5E   D0    00CD   610            MOVL    SP,R2                           ; save addr to calculate size
             00   DD    00D0   611            PUSHL   #00                             ; mark end of list
                        00D2   612
                        00D2   613    ;
                        00D2   614    ; Create item for devchar value
                        00D2   615    ;
                        00D2   616
             00   DD    00D2   617            PUSHL   #00                             ; return size
             69   DF    00D4   618            PUSHAL  IFB$L_PRIM_DEV(R9)              ; return buffer
    00020004 8F   DD    00D6   619            PUSHL   #<DVI$_DEVCHAR@16>+4            ; item code and buffer size
                        00DC   620
                        00DC   621    ;
                        00DC   622    ; Create item for secondary devchar value
                        00DC   623    ;
                        00DC   624
             00   DD    00DC   625            PUSHL   #00                             ; return size
        008C C9   DF    00DE   626            PUSHAL  IFB$L_AS_DEV(R9)                ; return buffer
    00030004 8F   DD    00E2   627            PUSHL   #<<DVI$_DEVCHAR!DVI$C_SECONDARY>@16>+4  ; item code and buffer size
                        00E8   628
                        00E8   629    ;
                        00E8   630    ; Create item for devbufsiz value
                        00E8   631    ;
                        00E8   632
             00   DD    00E8   633            PUSHL   #00                             ; return size
          48 A9   DF    00EA   634            PUSHAL  IFB$L_DEVBUFSIZ(R9)             ; return buffer
    00080004 8F   DD    00ED   635            PUSHL   #<DVI$_DEVBUFSIZ@16>+4          ; item code and buffer size
                        00F3   636
                        00F3   637    ;
                        00F3   638    ; Create item for secondary devbufsiz value
                        00F3   639    ;
                        00F3   640
             00   DD    00F3   641            PUSHL   #00                             ; return size
```

```
ST
Ps

PS
--
.
RM
$A

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
11
Th
11
37

Ma
--
_$
_$
_$
TO

24

Th

MA
```

```
    0094 C9    DF  00F5   642          PUSHAL  IFB$L_ASDEVBSIZ(R9)      ; return buffer
00090004 8F    DD  00F9   643          PUSHL   #<<DVI$_DEVBUFSIZ!DVI$C_SECONDARY>@16>+4; item code and buffer size
                   00FF   644
                   00FF   645 ;
                   00FF   646 ; Create item for devclass value
                   00FF   647 ;
                   00FF   648
      00       DD  00FF   649          PUSHL   #00                      ; return size
    1C AA      DF  0101   650          PUSHAL  FWA$L_DEV_CLASS(R10)     ; return buffer
00040004 8F    DD  0104   651          PUSHL   #<DVI$_DEVCLASS@16>+4    ; item code and buffer size
                   010A   652
                   010A   653 ;
                   010A   654 ; Create item for devnam string
                   010A   655 ;
                   010A   656
  0190 CA      3F  010A   657          PUSHAW  FWA$Q_SHRFIL(R10)        ; return size
  0194 CA      DD  010E   658          PUSHL   FWA$Q_SHRFIL+4(R10)      ; return buffer
00E80010 8F    DD  0112   659          PUSHL   #<DVI$_FULLDEVNAM@16>-
                   0118   660                  +FWA$S_SHRFILBUF         ; item code and buffer size
                   0118   661
                   0118   662 ;
                   0118   663 ; Create item for devlocknam string
                   0118   664 ;
                   0118   665
  0198 CA      3F  0118   666          PUSHAW  FWA$Q_SHRFIL_LCK(R10)    ; return size
  019C CA      DD  011C   667          PUSHL   FWA$Q_SHRFIL_LCK+4(R10)  ; return buffer
00F00010 8F    DD  0120   668          PUSHL   #<DVI$_DEVLOCKNAM@16>-
                   0126   669                  +FWA$S_SHRFIL_LCKNAM     ; item code and buffer size
                   0126   670
                   0126   671 ;
                   0126   672 ; Create item for secondary devnam string
                   0126   673 ;
                   0126   674
  01A0 CA      3F  0126   675          PUSHAW  FWA$Q_AS_SHRFIL(R10)     ; return size
  01A4 CA      DD  012A   676          PUSHL   FWA$Q_AS_SHRFIL+4(R10)   ; return buffer
00E90010 8F    DD  012E   677          PUSHL   #<<DVI$_FULLDEVNAM!DVI$C_SECONDARY>@16>-
                   0134   678                  +FWA$S_AS_SHRFILBUF      ; item code and buffer size
                   0134   679
                   0134   680 ;
                   0134   681 ; Create item for recsiz value
                   0134   682 ;
                   0134   683
      00       DD  0134   684          PUSHL   #00                      ; return size
    20 AA      DF  0136   685          PUSHAL  FWA$L_RECSIZ(R10)        ; return buffer
00180004 8F    DD  0139   686          PUSHL   #<DVI$_RECSIZ@16>+4      ; item code and buffer size
                   013F   687
                   013F   688 ;
                   013F   689 ; Create item for volume label
                   013F   690 ;
                   013F   691
      00       DD  013F   692          PUSHL   #00                      ; return size
  0920 CA      DF  0141   693          PUSHAL  FWA$T_VOLNAM(R10)        ; return buffer
0022000C 8F    DD  0145   694          PUSHL   #<DVI$_VOLNAM@16>+12     ; item code and buffer size
  51    5E     D0  014B   695          MOVL    SP,R1                    ; save addr for call
  5E    08     C2  014E   696          SUBL2   #8,SP                    ; make iosb
  50    5E     D0  0151   697          MOVL    SP,R0                    ; save addr for iosb
                   0154   698          $GETDVIW_S -
```

```
                0154                      CHAN = IFB$W_CHNL(R9),- ; I/O channel number
                0154                      EFN = #IMP$C_ASYQIOEFN,-; throw-away efn
                0154                      IOSB = (R0),-           ; iosb for synchronizing
                0154                      ITMLST = (R1)           ; item list
                016B
   5E   52  D0  016B            MOVL      R2,SP                            ; restore stack
     0F 50  E9  016E            BLBC      R0,ERRASGN1                      ; exit on error
                0171
                0171     ;
                0171     ; Subtract one from the device name sizes in order to get rid of the ":"
                0171     ; at the end of the name
                0171     ;
   0190 CA  B7  0171            DECW      FWA$Q_SHRFIL(R10)
   01A0 CA  B7  0175            DECW      FWA$Q_AS_SHRFIL(R10)
                0179
                0179            .IF       NDF,STASWITCH
                0179
                0179     ;
                0179     ; Deal with FOREIGN devices in a separate code section.  They all allow ANSI-"a"
                0179     ; names, so don't worry about checking them.
                0179     ;
                0179
                0179            BBS       #DEV$V_FOR,IFB$L_PRIM_DEV(R9),FOREIGN ; branch if dev mntd foreign
                0179            BBS       #FAB$V_NFS+FOP,(R8),FOREIGN     ; branch if non-file-struct.
                0179            BBS       #DEV$V_SQD,IFB$L_PRIM_DEV(R9),10$; branch if magtape
                0179            BBC       #DEV$V_DIR,IFB$L_PRIM_DEV(R9),10$; branch if not dir device
                0179
                0179     ;
                0179     ; At this point we are sure that we are not using an ANSI magtape device.
                0179     ; Check to see if we have an ANSI quoted string in the FWA, denoted by
                0179     ; FWA$V_QUOTED set and FWA$V_NODE not set.
                0179     ;
                0179
                0179            BBC       #FWA$V_QUOTED,(R10),10$ ; quoted not set
                0179            BBC       #FWA$V_NODE,(R10),20$   ; networking, no problem
                0179
                0179            .ENDC     ;STASWITCH
                0179
            05  0179     10$:   RSB                                        ; exit
                017A
                017A     20$:   RMSERR    FNM                              ; error in file name
            05  017F            RSB
                0180
                0180
                0180           .IF     NDF,STASWITCH
                0180   ;
                0180   ; User has requested non-file-structured operations (via the nfs fop bit)
                0180   ; or device was mounted foreign.
                0180   ;
                0180   ; Clear the 'dir'' and 'sdi' device characteristics (directory, single directory)
                0180   ; and set the 'for' device characteristic (foreign) and set up various other
                0180   ; file attributes.
                0180   ;
                0180
                0180   FOREIGN:
                0180           BICB2   #DEV$M_DIR!DEV$M_SDI,IFB$L_PRIM_DEV(R9); say no directory
                0180           SSB     #DEV$V_FOR,IFB$L_PRIM_DEV(R9); say foreign device
                0180           BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R9),CHKRND; branch if not magtape
                0180           MOVZWL  FWA$L_RECSIZ(R10),R1        ; fixed recsize from mount
                0180           BEQL    SETREC                      ; branch if not speced
                0180           BBC     #IFB$V_CREATE,(R9),10$      ; branch if doing open
                0180
                0180   ;
                0180   ;  This is create.  Only allow blocking if rfm=fix and mrs = recordsize.
                0180   ;
                0180
                0180           CMPB    FAB$B_RFM(R8),#FAB$C_FIX; rfm = fix?
                0180           BNEQ    SETREC                      ; branch if not
                0180           CMPW    FAB$W_MRS(R8),R1            ; same size?
                0180           BNEQ    SETREC                      ; branch if not
                0180   10$:    MOVW    R1,IFB$W_MRS(R9)            ; set fixed rec size from
                0180                                               ; mount parameter.
                0180           BICB2   #DEV$M_REC,IFB$L_PRIM_DEV(R9); clear 'unit rec' char.
                0180           BRB     SETFIX                      ; go set fixed record length
                0180   CHKRND: BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R9),SETREC; branch if not disk
                0180           MOVW    IFB$L_DEVBUFSIZ(R9),IFB$W_MRS(R9); say fixed 512 records
                0180           MNEGL   #2,IFB$L_EBK(R9)            ; say large eof blk
                0180                                               ; (RM1CREATE increments this)
                0180           MNEGL   #1,IFB$L_HBK(R9)            ; say large allocation
                0180   SETFIX: MOVB    #FAB$C_FIX,IFB$B_RFMORG(R9); say fixed records len.
                0180           MOVW    IFB$W_MRS(R9),IFB$W_LRL(R9); set fixed rec len.
                0180           BRB     EXIT                        ; continue
                0180   SETREC: BISB2   #DEV$M_REC,IFB$L_PRIM_DEV(R9); say mt is unit rec
                0180   EXIT:   BBS     #IFB$V_BIO,IFB$B_FAC(R9),10$; branch if block i/o
                0180           SSB     #IFB$V_BRO,IFB$B_FAC(R9); allow block i/o functions
                0180   10$:    RSB
                0180
                0180           .ENDC   ;STASWITCH
                0180
                0180   ERRASGN1:
FF18    31      0180           BRW     ERRASGN
```

```
0183
0183            .IF     NDF,STASWITCH
0183     ;
0183     ;  Process-Permanent File  -  check device characteristcs for terminal
0183     ;  and if so reassign the channel in super mode.
0183     ;
0183     ;  Also reset device characteristics to indicate get/put access
0183     ;  via dev.
0183     ;
0183
0183     DIRECTPPF:
0183            BBC     #DEV$V_TRM,IFB$L_PRIM_DEV(R9),SETDEV
0183
0183     ;
0183     ;  Process permanent file is a terminal  -  reassign channel in super mode
0183     ;  retain the exec mode channel until we have a super mode channel,
0183     ;  so that the virtual terminal ucb will not disappear.  This,
0183     ;  unfortunately, has the property that the user can get a spurious
0183     ;  no more channels error.
0183     ;
0183
0183            MOVW    IFB$W_CHNL(R9),R2           ; save exec channel number
0183            $ASSIGN_S       -
0183                    FWA$Q_DEVICE(R10),-
0183                    IFB$W_CHNL(R9),-
0183                    #PSL$C_SUPER
0183            MOVL    R0,R3                       ; preserve assign status code
0183            $DASSGN_S       R2                  ; return the exec mode channel
0183            BLBC    R3,ERRASGN1
0183
0183     ;
0183     ; Assume no change in dev chars; hence no need to reprocess.
0183     ;
0183     ;  If FAB$V_GET not set in fac, clear DEV$V_IDV.
0183     ;  If FAB$V_PUT not set in fac, clear DEV$V_ODV.
0183     ;
0183
0183     SETDEV: BBS     #FAB$V_GET,IFB$B_FAC(R9),10$; branch if 'get' on
0183            CSB     #DEV$V_IDV,IFB$L_PRIM_DEV(R9)
0183            CSB     #DEV$V_IDV,IFB$L_AS_DEV(R9)
0183
0183            ASSUME  FAB$V_PUT       EQ      0
0183
0183     10$:   BLBS    IFB$B_FAC(R9),20$           ; branch if 'put' on
0183            CSB     #DEV$V_ODV,IFB$L_PRIM_DEV(R9)
0183            CSB     #DEV$V_ODV,IFB$L_AS_DEV(R9)
0183     20$:   BRW     RM$SETFIB                   ; return to main line
0183
0183            .ENDC   ;STASWITCH
```

```
0183
0183          .IF     NDF,STASWITCH
0183     ;
0183     ;  Handle indirect open of process permanent file.
0183     ;
0183     ;  Perform various checks to see if operation possible.
0183     ;
0183     ;  (Note: This routine has the side effect of turning a $CREATE
0183     ;        on an inidrect process permanent file into an $OPEN)
0183     ;
0183
0183 INDPPF:
0183          $TSTPT  INDPPF
0183
0183          ASSUME  FAB$V_UFO          GE       16
0183          ASSUME  FAB$V_NFS          GE       16
0183
0183 ;
0183 ; Check for various invalid options.
0183 ;
0183
0183          BITB    #<FAB$M_UFO!FAB$M_NFS>@-16,FAB$L_FOP+2(R8)
0183          BNEQ    ERRFOP                         ; branch if any specified
0183          TSTB    FWA$B_ESCTYP(R10)              ; escape type 0?
0183          BNEQ    ERRLNE                         ; branch if not (not legal)
0183          CMPZV   #14,#2,FWA$W_ESCIFI(R10),#^B10; escape ifi indicate a ppf?
0183          BNEQ    ERRLNE                         ; branch if not
0183
0183     ;
0183     ;  Get address of process permanent file IFAB and check it out.
0183     ;
0183
0183          MOVL    R9,R7                          ; save ifab addr
0183
0183          ASSUME  FAB$V_PPF_IND    GE       8
0183
0183          BISB2   #FAB$M_PPF_IND@-8,FWA$W_ESCIFI+1(R10); get non-privileged ppf ifi
0183          MOVZWL  FWA$W_ESCIFI(R10),R9           ; and move it to r9
0183          MOVL    #IMP$C_IFABTBL/4,R0            ; ifab table offset/4
0183          PUSHL   R11                            ; save impure area pointer
0183          BSBW    RM$GTIADR                      ; get ifab address
0183          POPR    #^M<R11>                       ; restore image i/o ptr (save cc's)
0183          BEQL    ERRIFI                         ; branch if no ppf ifab
0183          CMPB    IFB$B_BID(R9),#IFB$C_BID; is it an ifab?
0183          BNEQ    ERRBG3                         ; really bad if not!
0183
0183     ;
0183     ;  Got ppf IFAB o.k.
0183     ;
0183     ;  Insure org is sequential and rfm is not UDF
0183     ;  (Note: User fac is ignored.  Each operation will be checked.)
0183     ;
0183
0183          ASSUME  FAB$C_SEQ          EQ       0
0183
0183          TSTB    IFB$B_ORGCASE(R9)              ; seq file org?
0183          BNEQ    ERRORG                         ; branch if not (not supported)
```

```
0183
0183                    ASSUME  FAB$C_UDF       EQ      0
0183
0183                    TSTB    IFB$B_RFMORG(R9)                ; undefined format?
0183                    BEQL    ERRRFM                          ; branch if so (not supported)
0183
0183         ;
0183         ;  Check for valid rat value: must be only one of cr, ftn, or prn.
0183         ;  If prn set in rat must also be set for real file.
0183         ;
0183         ;  Isolate carriage control bits
0183         ;
0183                    CLRL    R1                              ; zero-extend the field
0183                    BICB3   #255\<FAB$M_CR!FAB$M_FTN!FAB$M_PRN>,FAB$B_RAT(R8),R1
0183                    BBC     #DEV$V_ODV,IFB$L_PRIM_DEV(R9),5$; branch if input only
0183                    BBC     #FAB$V_PRN,R1,5$                ; branch if not prn
0183                    BBC     #FAB$V_PRN,IFB$B_RAT(R9),ERRRAT; branch if ppf not prn
0183
0183                    ASSUME  FAB$V_FTN EQ 0
0183                    ASSUME  FAB$V_CR EQ FAB$V_FTN+1
0183                    ASSUME  FAB$V_PRN EQ FAB$V_CR+1
0183
0183         5$:        BBC     R1,I^#^B00010111,ERRRAT ; branch if more than 1 ccl attr
0183
0183         ;
0183         ;  Check for ppf IFAB busy setting busy if not in use.
0183         ;
0183
0183         10$:       BBCS    #IFB$V_BUSY,(R9),IND_PPF_OK; branch if not busy
0183
0183
0183         ERRFOP:
0183                    RMSERR  FOP                             ; invalid fop bits for ppf
0183                    RSB
0183
0183         ERRLNE:
0183                    RMSERR  LNE                             ; invalid equivalence string
0183                    RSB
0183
0183         ERRBG3:    RMSTBUG FTL$_BADIFAB                    ; invalid ifab
0183
0183         ERRIFI:
0183                    RMSERR  IFI                             ; invalid equiv. string ifi
0183                    BRB     RESTORE_IFAB
0183
0183         ERRORG:
0183                    RMSERR  ORG                             ; ppf not sequential org
0183                    BRB     RESTORE_IFAB
0183
0183         ERRRFM:
0183                    RMSERR  RFM                             ; ppf UDF format
0183                    BRB     RESTORE_IFAB
0183
0183         ERRRAT:
0183                    RMSERR  RAT                             ; bad rat value
0183         RESTORE_IFAB:
0183                    MOVL    R7,R9                           ; restore temp. ifab addr
```

```
        0183   956              RSB                                     ; go handle error
        0183   957
        0183   958
        0183   959 ERRNAM:
        0183   960              POPR    #^M<R1>                         ; get ppf ifab addr
        0183   961              CSB     #IFB$V_BUSY,(R1)                ; clear busy
        0183   962              RSB                                     ; go clean up
        0183   963
```

```
0183   965
0183   966 ;
0183   967 ;  The indirect open of the process permanent file is permissable.
0183   968 ;  switch to using the ppf IFAB for the remainder of this open.
0183   969 ;
0183   970 ;  Must perform the following:
0183   971 ;
0183   972 ;          1. Set ppf_image bit.
0183   973 ;          2. Copy caller's mode and arg list address to the ppf ifab.
0183   974 ;             (Note: there will be no asb because no stall can have occurred.)
0183   975 ;          3. Return the temporary ifab and fwa.
0183   976 ;          4. Zero the ifi table entry to temporary ifab.
0183   977 ;          5. Move address of pio impure area to r11 and copy saved sp
0183   978 ;             to pio impure area.
0183   979 ;
0183   980
0183   981 IND_PPF_OK:
0183   982         SSB     #IFB$V_PPF_IMAGE,(R9)     ; indicate indirect processing
0183   983                                           ;  of ppf
0183   984         MOVB    IFB$B_MODE(R7),IFB$B_MODE(R9); save caller's mode
0183   985         MOVL    IFB$L_ARGLST(R7),IFB$L_ARGLST(R9); and arg list addr
0183   986         PUSHL   R9                        ; save ppf ifab addr
0183   987
0183   988 ;
0183   989 ;  Fill in name block, if any, with resultant string and dvi.
0183   990 ;
0183   991
0183   992         MOVL    R7,R9                     ; restore temp ifab addr
0183   993         BSBW    RM$FILLNAM                ; fill in nam blk if any
0183   994         BLBC    R0,ERRNAM                 ; branch on error
0183   995         MOVW    FWA$W_ESCIFI(R10),AP      ; save ifi code
0183   996         BSBW    RM$DEALLOCATE_FWA         ; deallocate fwa and associated struct.
0183   997         MOVL    R9,R3                     ; ifab addr to r3,
0183   998         MOVL    R9,R4                     ;  r4 for retblk
0183   999         BSBW    RM$RETBLK                 ; deallocate ifab
0183  1000
0183  1001 ;
0183  1002 ; return all pages used
0183  1003 ;
0183  1004
0183  1005         SUBL3   #8,R9,R10                 ; get ifab list head
0183  1006         MOVL    (R10),R4                  ; get first hole (must be there - ifb)
0183  1007
0183  1008 1$:     PUSHL   (R4)                      ; save forward pointer
0183  1009         CMPL    8(R4),#504                ; hole should be at least a page (-header)
0183  1010         BLSSU   DEALLERR                  ; some one goofed
0183  1011         BSBW    RM$RET1PAG                ; return it
0183  1012         POPL    R4                        ; restore pointer to next hole
0183  1013         CMPL    R4,R10                    ; all done?
0183  1014         BNEQ    1$                        ; do next
0183  1015
0183  1016         BSBW    RM$ZAPIFI                 ; zero ifab table entry
0183  1017         MOVL    (SP)+,R9                  ; restore ppf ifab addr
0183  1018         MOVL    R9,R10                    ; copy to r10
0183  1019
0183  1020 ;
0183  1021 ; Copy saved sp value.
```

```
                           0183  1022 ;
                           0183  1023
                           0183  1024         MOVL    IMP$L_SAVED_SP(R11),@#PIO$GW_PIOIMPA+IMP$L_SAVED_SP
                           0183  1025         MOVAW   @#PIO$GW_PIOIMPA,R11     ; get pio impure area addr
                           0183  1026
                           0183  1027 ;
                           0183  1028 ;  Output the new ifi value and bls field to fab.
                           0183  1029 ;
                           0183  1030
                           0183  1031         MOVW    IFB$L_ASDEVBSIZ(R9),FAB$W_BLS(R8); default buffer size
                           0183  1032         POPL    R0                          ; pop return pc
                           0183  1033
                           0183  1034 ;
                           0183  1035 ;  If user specified stream format but RAT=0, force RAT=CR
                           0183  1036 ;  to get proper output.
                           0183  1037 ;
                           0183  1038
                           0183  1039         CMPB    FAB$B_RFM(R8),#FAB$C_STM ; user specified stream format?
                           0183  1040         BLSSU   5$                          ; branch if not
                           0183  1041         BITB    #<FAB$M_CR!FAB$M_FTN!FAB$M_PRN>,-
                           0183  1042                 FAB$B_RAT(R8)               ; user specified RAT?
                           0183  1043         BNEQ    5$                          ; branch if so
                           0183  1044         SSB     #FAB$V_CR,FAB$B_RAT(R8)  ; force RAT=CR
                           0183  1045 5$:     INSV    FAB$B_RAT(R8),#FAB$V_PPF_RAT,#FAB$S_PPF_RAT,AP; save rat in ifi
                           0183  1046         MOVW    AP,FAB$W_IFI(R8)            ; construct non-priv. ifi
                           0183  1047
                           0183  1048 ;
                           0183  1049 ;  Return the real rat and rfm values for the file, except
                           0183  1050 ;  If this is a unit record device and the user has not specified
                           0183  1051 ;  print file carriage control, return rfm=var and don't change the
                           0183  1052 ;  user's rat field.
                           0183  1053 ;
                           0183  1054
                           0183  1055         ASSUME  DEV$V_REC EQ 0
                           0183  1056
                           0183  1057         BLBS    IFB$L_PRIM_DEV(R9),20$   ; branch if unit record
                           0183  1058 10$:    BRW     RM$COPRTN1                  ; go finish up open
                           0183  1059 20$:    BBS     #FAB$V_PRN+FAB$V_PPF_RAT,AP,10$; branch if user specifed
                           0183  1060                                             ; print file carriage ctl.
                           0183  1061         MOVB    #FAB$C_VAR,FAB$B_RFM(R8); change rfm to var
                           0183  1062         BRW     RM$COPRTN2                  ; finish up open without
                           0183  1063                                             ;  altering user's rat
                           0183  1064
                           0183  1065 DEALLERR:
                           0183  1066         RMSTBUG FTL$_DEALLERR
                           0183  1067
                           0183  1068         .ENDC   ;STASWITCH
```

```
_$

Sy
--
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
MT
MT
MT
MT
MT
MT
MT
MT
MT
MT
OT
OT
OT
```

```
                             0183        .SBTTL  RM$RET_DEV_CHAR - Set device characteristics into FAB
                             0183 ;++
                             0183 ;
                             0183 ;  Set the associated device characteristics into the FAB.
                             0183 ;
                             0183 ;  The following algorithm is used to determine the characteristics:
                             0183 ;
                             0183 ;      Foreign-mounted devices:
                             0183 ;              FAB$L_DEV = FAB$L_SDC = IFB$L_AS_DEV
                             0183 ;      Spooled devices:
                             0183 ;              FAB$L_DEV = IFB$L_AS_DEV
                             0183 ;              FAB$L_SDC = IFB$L_PRIM_DEV
                             0183 ;      Network devices:
                             0183 ;              As determined by NT$REG_DEV_CHAR
                             0183 ;      Indirect PPF sequential devices:
                             0183 ;              FAB$L_DEV = IFB$L_PRIM_DEV modified to be a record device:
                             0183 ;                      DIR, FOD, RND, SDI, SQD  cleared
                             0183 ;                      REC, CCL set
                             0183 ;              FAB$L_SDC = IFB$L_AS_DEV
                             0183 ;      All other normal devices:
                             0183 ;              FAB$L_DEV = IFB$L_PRIM_DEV
                             0183 ;              FAB$L_SDC = IFB$L_AS_DEV
                             0183 ;
                             0183 ; If device is foreign, move AS_DEV characteristics into both FAB
                             0183 ; characteristics words, since the PRIM_DEV has been altered for
                             0183 ; internal RMS processing requirements (e.g. DIR has been cleared, FOR
                             0183 ; may have been set because of NFS, etc.).
                             0183 ;
                             0183 ;--
                             0183
                             0183 RM$RET_DEV_CHAR::
                             0183
        40 A8    69     D0   0183        MOVL    IFB$L_PRIM_DEV(R9),FAB$L_DEV(R8)         ; set prim dev char
     44 A8    008C C9   D0   0187        MOVL    IFB$L_AS_DEV(R9),FAB$L_SDC(R8)           ; set secondary dev char
        0A 69    18     E0   018D        BBS     #DEV$V_FOR,IFB$L_PRIM_DEV(R9),10$         ; diff if foreign
   0A 008C C9    06     E1   0191        BBC     #DEV$V_SPL,IFB$L_AS_DEV(R9),20$           ; okay if not spooled
        44 A8    69     D0   0197        MOVL    IFB$L_PRIM_DEV(R9),FAB$L_SDC(R8)         ; set alt dev. char.
  40 A8   008C C9       D0   019B 10$:   MOVL    IFB$L_AS_DEV(R9),FAB$L_DEV(R8)           ; set alt dev. char.
                             01A1
                             01A1        ASSUME  FAB$C_SEQ       EQ      0
                             01A1
           23 A9 95          01A1 20$:   TSTB    IFB$B_ORGCASE(R9)                        ; sequential file org?
              10 12          01A4        BNEQ    30$                                      ; branch if not
                             01A6
                             01A6 ;
                             01A6 ;  If this is an indirect process-permanent file, set up the device characteristics
                             01A6 ;  to make it look like a unit record device.
                             01A6 ;
                             01A6
        0C 69    22     E1   01A6        BBC     #IFB$V_PPF_IMAGE,(R9),30$                ; branch if not indirect ppf
  40 A8   10004038 8F   CA   01AA        BICL2   #DEV$M_DIR!DEV$M_FOD!DEV$M_RND -         ; clear file characteristics
                             01B2                !DEV$M_SDI!DEV$M_SQD,FAB$L_DEV(R8)
                 03     C8   01B2        BISL2   #DEV$M_REC!DEV$M_CCL,-                   ; and set unit record chars.
              40 A8          01B4                FAB$L_DEV(R8)
                             01B6
                             01B6 ;
                             01B6 ;  Network specific code
        01B6  ;
        01B6
        01B6  30$:
        01B6             .IF     NDF,STASWITCH
        01B6
        01B6             BBC     #IFB$V_DAP,(R9),40$                         ; branch if not network oper
        01B6             CLRB    IFB$B_ORGCASE(R9)                           ; zero orgcase to subsequent
        01B6                                                                 ;   force rel and idx file op
        01B6                                                                 ;   thru sequential code!!!
        01B6             JSB     NT$RET_DEV_CHAR                             ; return real device char to
        01B6
        01B6             .ENDC   ;STASWITCH
        01B6                                                                 ;   if they were returned by
  05    01B6  40$:       RSB
        01B7
        01B7
        01B7             .END
```

```
$$.PSECT_EP                   = 00000000                FWA$Q_SHRFIL            = 00000190
$$ARGS                        = 00000004                FWA$Q_SHRFIL_LCK        = 00000198
$$RMSTEST                     = 0000001A                FWA$S_AS_SHRFILBUF      = 00000010
$$RMS_PBUGCHK                 = 00000010                FWA$S_SHRFILBUF         = 00000010
$$RMS_TBUGCHK                 = 00000008                FWA$S_SHRFIL_LCKNAM     = 00000010
$$RMS_UMODE                   = 00000004                FWA$T_VOLNAM            = 00000920
$$T1                          = 00000001                FWA$V_CONCEAL_DEV       = 00000039
ASSIGN                          00000000 R      01      GETDEV_CHAR               000000CD R      01
ASSIGN$_ACMODE                = 0000000C                IFB$B_MODE              = 0000000A
ASSIGN$_CHAN                  = 00000008                IFB$B_ORGCASE           = 00000023
ASSIGN$_DEVNAM                = 00000004                IFB$L_ASDEVBSIZ         = 00000094
ASSIGN$_MBXNAM                = 00000010                IFB$L_AS_DEV            = 0000008C
ASSIGN$_NARGS                 = 00000004                IFB$L_DEVBUFSIZ         = 00000048
CCB$L_UCB                     = 00000000                IFB$L_LAST_FAB          = 00000024
DC$_DISK                      = 00000001                IFB$L_PRIM_DEV          = 00000000
DC$_TAPE                      = 00000002                IFB$V_PPF_IMAGE         = 00000022
DEV$M_CCL                     = 00000002                IFB$W_CHNL              = 00000020
DEV$M_DIR                     = 00000008                IMP$C_ASYQIOEFN         = 0000001F
DEV$M_FOD                     = 00004000                IO$_PACKACK               ********  X     01
DEV$M_FOR                     = 01000000                IOC$VERIFYCHAN            ********  X     01
DEV$M_MNT                     = 00080000                PSL$C_EXEC              = 00000001
DEV$M_REC                     = 00000001                RM$ASSIGN                 00000000 RG     01
DEV$M_RND                     = 10000000                RM$RET_DEV_CHAR           00000183 RG     01
DEV$M_SDI                     = 00000010                RMS$_CHN                = 0001C0EC
DEV$M_SQD                     = 00000020                RMS$_FNM                = 0001852C
DEV$V_FOR                     = 00000018                STASWITCH               = 00000001
DEV$V_SPL                     = 00000006                STA_VOL_VAL               000000A9 R      01
DVI$C_SECONDARY               = 00000001                SYS$ASSIGN                ********  X     01
DVI$_DEVBUFSIZ                = 00000008                SYS$CMKRNL                ******** GX     01
DVI$_DEVCHAR                  = 00000002                SYS$DASSGN                ******** GX     01
DVI$_DEVCLASS                 = 00000004                SYS$GETDVIW               ******** GX     01
DVI$_DEVLOCKNAM               = 000000F0                SYS$QIOW                  ******** GX     01
DVI$_FULLDEVNAM               = 000000E8                UCB$L_DEVCHAR           = 00000038
DVI$_RECSIZ                   = 00000018                UCB$M_VALID             = 00000800
DVI$_VOLNAM                   = 00000022                UCB$W_STS               = 00000064
ERRASGN                         0000009B R      01
ERRASGN1                        00000180 R      01
EXE$GL_FLAGS                    ********  X     01
EXE$V_INIT                      ********  X     01
FAB$B_ACMODES                 = 0000004A
FAB$C_SEQ                     = 00000000
FAB$L_DEV                     = 00000040
FAB$L_FOP                     = 00000004
FAB$L_SDC                     = 00000044
FAB$L_STV                     = 0000000C
FAB$M_NFS                     = 00010000
FAB$M_UFO                     = 00020000
FAB$S_CHAN_MODE               = 00000002
FAB$V_CHAN_MODE               = 00000002
FAB$V_NFS                     = 00000010
FAB$V_UFO                     = 00000011
FOP                           = 00000020
FWA$L_DEV_CLASS               = 0000001C
FWA$L_RECSIZ                  = 00000020
FWA$Q_AS_SHRFIL               = 000001A0
FWA$Q_CONCEAL_DEV             = 000000E8
FWA$Q_DEVICE                  = 000000E0
```

```
                                             +----------------+
                                             ! Psect synopsis !
                                             +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
RMS$RMS0                         000001B7  (  439.)  01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD  NOWRT NOVEC BYTE
$ABS$                            00000000  (    0.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                             +------------------------+
                                             ! Performance indicators !
                                             +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       40    00:00:00.08    00:00:00.84
Command processing                  138    00:00:01.03    00:00:05.90
Pass 1                              503    00:00:20.45    00:00:55.01
Symbol table sort                     0    00:00:03.25    00:00:03.90
Pass 2                              193    00:00:04.53    00:00:11.33
Symbol table output                  13    00:00:00.13    00:00:00.15
Psect synopsis output                 2    00:00:00.02    00:00:00.05
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals                891    00:00:29.49    00:01:17.18

The working set limit was 1950 pages.
113895 bytes (223 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2215 non-local and 14 local symbols.
1145 source lines were read in Pass 1, producing 14 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.

                                             +--------------------------+
                                             ! Macro library statistics !
                                             +--------------------------+

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                           10
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                            3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        19
TOTALS (all libraries)                                   32

2440 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:STAPRFLNM/OBJ=OBJ$:STAPRFLNM MSRC$:STASW/UPDATE=(ENH$:STASW)+MSRC$:RMOPRFLNM/UPDATE=(ENH$:RMOPRFLNM)+EXECML$/LIB+LIB$
```

0331 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMS0TRUNC
LIS

STAPRFLNM
LIS

RPGCVTPTO
LIS

RPGHANDLE
LIS

RPGMOVE1
LIS

RPGLIB
REQ

RMSGBL
LIS

RPGRTL

RPGDSPLY
LIS

RPGRTL
MAP

RPGPROLOG
REQ

RPGEXTIND
LIS

RPGLIB
LIS

RMS0SRCH
LIS

RMS0WAIT
LIS

RPGBTZ
LIS

RPGMOVE2
LIS

RPGDIVIDE
LIS

RPGIOEXCE
LIS

RPGDEF
REQ

RMS0UPDAT
LIS

RPGERROR
LIS